Placement of Software-as-a-Service Components in Cloud Computing Environment

Alok Kumar

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India
June 2014

Placement of Software-as-a-Service Components in Cloud Computing Environment

Thesis submitted in partial fulfilment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Software Engineering)

by

Alok Kumar

(Roll No.- 212CS3122)

under the supervision of

Dr. Bibhudatta Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769 008, India

June 2014

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

Certificate

This is to certify that the work in the thesis entitled ***Placement of Software-as-a-Service Components in Cloud Computing Environment*** by ***Alok Kumar*** is a record of an original research work carried out by him under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Master of Technology with the specialization of Software Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Place: NIT Rourkela
Date: May 30, 2014

(Dr. Bibhudatta Sahoo)
Professor, CSE Department
NIT Rourkela, Odisha

This thesis is dedicated to
my Parents and Siblings

Acknowledgment

I owe deep gratitude to the ones who have contributed greatly in completion of this thesis. Foremost, I would like to express my sincere gratitude to my supervisor Prof. Bibhudatta Sahoo for providing me with a platform to work on challenging areas of placement of SaaS in cloud computing. His valuable comments and suggestions were encouraging. He was the one who showed me the path from the beginning to end.

I am also thankful to Prof. Santanu Kumar Rath, Prof. Sanjay Kumar Jena, Prof. Banshidhar Majhi, Prof. Durga Prasad Mohapatra, Prof. Ashok Kumar Turuk, Prof. Pabitra Mohan Khilar and Prof. Manmath Narayan Sahoo for giving encouragement and sharing their knowledge during my thesis work.

I must acknowledge the academic resources that I have got from NIT Rourkela. I would like to thank administrative and technical staff members of the Department of Computer Science and Engineering who have been kind enough to help us in their respective roles.

I would like to thank all my friends, lab mates and research scholars for their encouragement and understanding. They made my life beautiful and helped me every time when I was in some problem.

Most importantly, none of this would have been possible without the love and patience of my family. My family, to whom this thesis is dedicated to, has been a constant source of love, concern, support and strength all these years. I would like to express my heart-felt gratitude to them.

Alok Kumar

Roll-212cs3122

Abstract

Cloud computing is an emerging paradigm in which information technology resources are provided over the internet as a service to users. Software-as-a-Service (SaaS) is offered by cloud, which can be delivered in a composite form, consisting of a set of application and data components, that works together to deliver higher-level functional software. SaaS components are placed on top of the virtual machines (VMs) in cloud computing environment, which are deployed on physical or storage servers. The SaaS placement is an NP-hard problem. The research problem refers to how a SaaS component is placed on virtual machine to optimize its performance while satisfying the SaaS resource and response time constraints with service level agreement (SLA) constraints. This thesis presents SaaS placement problem as an optimization problem, to maximize the profit of the SaaS providers. Intractability nature of the SaaS placement problem leads to the use of genetic algorithms to obtain sub-optimal solution for SaaS component placement on virtual machine. A suitable codification scheme for SaaS component placement has been proposed for the genetic algorithm. The performance of proposed genetic algorithm has been compared with first-fit randomized algorithm (First-fit RA) by varying number of virtual machines and SaaS components by using in-house simulator. Performance of proposed genetic algorithm has been found to be better in comparison to First-fit RA.

Keywords: Cloud Computing; Software-as-a-Service; Cloud Modelling; SaaS Modelling; SaaS Component Placement based on Genetic Algorithm

Contents

Certificate ii

Dedication iii

Acknowledgement iv

Abstract v

List of Figures ix

List of Tables x

Abbreviation xi

1 Introduction 1

1.1 Introduction 1

1.2 Software as a Service Component 2

1.3 Related work 3

1.4 Research Motivation 5

1.5 Problem Statement 5

1.6 Research Contribution 6

1.7 Thesis Layout 6

Abbreviation 1

2 Cloud Infrastructure for SaaS Placement 7

2.1 Introduction 7

2.2 Cloud Computing 7

2.2.1 Cloud Architecture 8

2.2.2 Cloud Service Models 9

2.2.3 Cloud Deployment Models 9

2.3 Cloud Infrastructure Model 10
2.4 Cloud software as a service (SaaS) 10
2.4.1 Characteristics of SaaS 11
2.4.2 SaaS Examples 12
2.4.3 SaaS Component Model 12
2.5 Service Level Agreement 13
2.6 SaaS deployment constraints 14
2.6.1 Resource constraints: 14
2.6.2 Placement Constraints: 14
2.6.3 Execution time constraints: 14
2.6.4 Sequence of migration constraints: 15
2.6.5 Cost Constraints 15
2.7 Problem Formulation 16
2.8 Current State of Art of CPP 19
2.9 Summary 20
3 GA Framework for SaaS Component Placement 22
3.1 Introduction 22
3.2 Genetic Algorithms 23
3.2.1 GA Parameters 24
3.3 SCPGA 25
3.3.1 SCPGA Encoding 26
3.3.2 Infeasible Encoding Problem 27
3.3.3 Genetic Operators 27
3.3.4 Decision of Stopping Criteria 30
3.4 Results 33
3.4.1 Profit Finding w.r.t. Number of Virtual Machines 35
3.4.2 Profit Finding w.r.t. Number of SaaS Components 37
3.5 Summary 40
4 Conclusions 41
4.1 Conclusions 41

4.2 Future Work 41

Bibliography **42**

List of Figures

2.1 NIST Cloud Computing Reference Architecture (CCRA) [5] 9
2.2 Cloud Stack 9
2.3 SaaS component model in cloud computing infrastructure [58] . . . 13

3.1 Encoding schema of the SCPGA 27
3.2 Crossover operation in SCPGA 29
3.3 Mutation operation in SCPGA 30
3.4 Profit finding w.r.t. number of iterations 33
3.5 Data set screen shot-1 34
3.6 Data set screen shot-2 34
3.7 Profit finding via SCPGA and Firstfit RA with fixed number of SaaS components 37
3.8 Profit finding via SCPGA and Firstfit RA with fixed number of VMs 40

List of Tables

2.1 Sets and Attributes of Physical Resources in Cloud 17
2.2 Sets, Parameters and Requirements of Software as a Service 18
2.3 State of Art of SaaS Placement 20

3.1 Genetic Algorithmic Parameter for Profit Finding w.r.t. Number of Iteration 31
3.2 Profit finding via GA with fixed number of SaaS components and VMs for deciding the stopping criteria 32
3.3 Genetic Algorithmic Parameter for Profit Finding w.r.t. Number of Virtual Machines 35
3.4 Profit finding via SCPGA and Firstfit RA with fixed number of SaaS components 36
3.5 Genetic Algorithmic Parameter for Profit Finding w.r.t. Number of SaaS Components 38
3.6 Profit finding via SCPGA and Firstfit RA with fixed number of SaaS components 39

List of Abbreviations

NIST National Institute of Standard and Technology

VPN Virtual Private Network

IDC International Data Corporation

SLA Service Level Agreement

SaaS Software-as-a-Service

IaaS Infrastructure-as-a-Service

PaaS Platform-as-a-Service

CaaS Communication-as-a-Service

NaaS Network-as-a-Service

AC Application Component

BC Business Component

IC Integration Component

SC Storage Component

CPP Component Placement Problem

CSRSGA Cost-aware Service Request Scheduling based on Genetic Algorithm

NP Non-Probabilistic

VM Virtual Machine

SPLE Software Product Line Engineering

SOA Service Oriented Architecture

GA Genetic Algorithm

RA Randomized Algorithm

CCRA Cloud Computing Reference Architecture

ASP Application Service Provider

w.r.t. with respect to

SCPGA SaaS Component Placement based on Genetic Algorithm

Chapter 1

Introduction

1.1 Introduction

Currently, the need of computing paradigm or utility computing has increased up to where the IT transformation will have done. Cloud computing [7] is a computing paradigm, in which applications, data and IT resources are provided by vendors as a service to users over the broadband network. Cloud computing provides hardware services, infrastructure services, platform services, software services, storage services to different Internet applications. NIST defined cloud as: "cloud computing is a model which enables the services to the users. These services are convenient and on-demand or you can say available anywhere through the network as pay-per-use basis" [26].

Cloud services are categorized by NIST [26] in five different categories. 1) Software as a service (SaaS), offers software services to the users on pay-per-use basis. For example MS office 365, Salseforce.com, Google Docs etc. 2) Infrastructure as a service (IaaS), offers resources(processing, storage, networks and other fundamental computing resources)for computing, which are provided by cloud vendors to the service provider or users. For example Amazon EC2. 3) Communications as a service (CaaS), real-time communication and collaboration service capability provided to the cloud service user. 4) Cloud platform as a service (PaaS), provides a platform to the cloud service user is to deploy user developed applications onto the cloud infrastructure using platform tools like .NET, supported by the cloud service provider. 5) Network as a Service (NaaS), provides capability to the

cloud user, is to use transport connectivity services and/or inter-cloud network connectivity services like bandwidth, VPN etc.

SaaS has received most of the attention of IT industries. NIST defines SaaS as a category of cloud services where the capability provided to the cloud service user is to use the cloud service providers applications running on a cloud infrastructure. SaaS is an application model which provides software via Internet[54]. There are three levels [60] in the cloud: hardware, system, and application level. SaaS fill in the application level, which provide specific services to the end users.

Now a day, SaaS receives a lot of attention of software service providers. Software users also benefit from adaption of SaaS. A report from International Data Corporation (IDC) states that a significant increase in companies' subscriptions will happen due to SaaS practices in future. SaaS market increases every moment [8] and in 2007 Dubey and Wagle [20] reported that within three years the IT company's revenue increased by 18 percent due to adoption of SaaS. IDC reported that in 2009, the worldwide revenue for SaaS was $13.1 billion, and it will reach $45 billion till 2014 [15]. Gartner also forecasted that total revenue would reach $22.1 billion by 2015 [31].

The SaaS deployment are the installation to delivery of software services in cloud computing infrastructure. SaaS deployment is initiated by a cloud service provider via a user requesting process, which is generally automated. Alternately, SaaS deployment can be initiated by a third party managed (hosted) service provider. SaaS deployment is considered complete once a user has the necessary means to access a SaaS offering, regardless of whether or not the consumer begins using the service at the time it is defined in SLA.

1.2 Software as a Service Component

SaaS is a combination of different type of components; application component (AC), integration component (IC), business component (BC), and storage component (SC) [56] [59]. These components of SaaS represent in tuple form as S(AC, IC, BC, SC). Each SaaS component AC, IC, BC, and SC have some resource require-

ments. SLA specified the maximum response time of the SaaS. SaaS components are deployed on top of the virtual machines in cloud computing infrastructure which are provided by cloud vendors to the SaaS provider. A cloud data center consists two types of servers; computation and storage servers. Each server has some limited processing capacity, memory size and I/O capacity as well. Several types of VMs deployed to each physical server.

1.3 Related work

Component placement problem (CPP) is divided into two categories: 1) online CPP, refers placement problems of short-lived functional components and solved at runtime of the system [38] and 2) offline CPP, placement of the application is made in the beginning and takes few minutes or hours [61]. Existing research formulated CPP as a resource and cost optimization problem [37] [39] [40] [43] [46] [57], and as a candidate of multiple knapsack problem [52].

A. Karve et al. [37], define application instance placement on a given set of server machines to satisfy the resource requirement of each application cluster. Their objective was to with maximization of resource demand, to minimize the number of placement changes. For this purpose they proposed a middleware clustering technology capable of dynamically allocating resources to web applications through dynamic application instance placement. This paper also mentioned that the placement of SaaS is NP-hard problem.

Zhu et al. [61] addressed the placement problem not only for application's constraints as well as for storage requirement of the components. The location of the storage of component is already known before the placement process.

Zimmerova et al. [62] proposed placement method which focused communication between the components. This communication is captured using automata language and the placement is based on cost of interaction or communication between components.

Kichkaylo et al. [39] proposed a placement method similar to Zimmerova, where the application is defined by its components.

Urgaonkar et al. [52] used first-fit approximation algorithm for placement of component in offline CPP. This algorithm placed the component at the first server found that can satisfy requirements of a component.

Thomas Kwok et al. [40] tried to maximize cost savings and minimize the number of servers used. They calculated the resource requirement for multi-tenants with applied constraints in a shared application instance, and then found the optimal placement of tenants and instances with maximum cost saving without violating any requirements of SLA for all tenants in a set of servers.

Zerath Izzah Mohd et al. [57] find out the new placement in the dynamic nature of the workload in the cloud for composite services which are placed on VM. For this they had clustered the application components, such that new placement will minimize the resources' cost while satisfying the all quality parameters. To solve the problem they used a genetic algorithm with some modification.

Zerath Izzah Mohd et al. [58] presented the problem of composite SaaS resource management in the cloud. They mainly concerned about the initial placement and resource optimization problems to improve the SaaS performance based on its execution time as well as minimized the resource uses. They focus on the SaaS requirements, constraints and inter-dependencies.For this they had used evolutionary algorithms.

Moens et al. [46], defined a feature-based cloud resource management model, used software product line engineering (SPLE), in which products are composition of feature instances. These feature instances are developed by service-oriented architecture. They used the feature-based model from existing services. They used feature based instances in place of application instances, to increase the achievable level of multi-tenancy. For the placement purpose they had used meta heuristic algorithms.

Zhipiao Liu et al. [43] proposed a cloud service request model with consideration of SLA constraints, and present a cost-aware service request scheduling approach which was based on genetic algorithm (CSRSGA). Their approach lease and reuse virtual resources as well as minimize the rental cost of overall infras-

tructure for maximizing SaaS providers' profits. They adopt GA for achieving optimised request scheduling by including the heterogeneity of VMs in terms of their performance, configuration and pricing.

1.4 Research Motivation

A SaaS delivered as a composite application or as multiple components form, in which the software components are loosely coupled and components communicate to each other in order to provide a high-level functional system [23]. To provide the services via SaaS first there is a requirement to place the SaaS components on servers. The SaaS placement is an NP-hard problem [36] [37] [40] [44] [56]. So the overall aim of the research is to develop an efficient, scalable and convenient model to deliver the SaaS components to provide the services to the users with minimum revenue cost. Due to NP-hard nature of the problem, the solution of the problem via conventional algorithms is not possible, hence some heuristic algorithms are required to solve the problem and provide a sub-optimal solution which is very near to an optimal result. Most of the researchers used GA to solve this problem. This research also used Genetic Algorithm (GA) to handle the problems' challenges. GA is a stochastic search terminology which applies biological evolutions or operators in production of solution [25]. GA have been applied in many problems which are complex, large-scaled, constrained and optimizations in domains like web services, engineering and technology [9]. These are the main reasons due to which GA was chosen to solve the problem.

1.5 Problem Statement

As mentioned in previous sections, this research focus upon placement of SaaS with SLA constraints and resource constraints. In this problem a set of task will carry out that will place the SaaS component on suitable VMs. First step in SaaS lifecycle is the placement of SaaS on one or more VMs available in cloud computing environment. Placement of SaaS carried out during the initial phase of deployment of SaaS. A composite SaaS deployed in cloud is composed of several type

of components like, application component, data component, integration components etc. The problem addressed in this thesis is placement of SaaS components on VMs to maximize the profit. The optimal placement for SaaS components on VMs are to be carried out such that the performance of SaaS is optimal, while satisfying SLA constraints and resource constraints. The placement of SaaS components are also maximizes the profit of the SaaS provider by minimizing resource cost, capital expenditure of users, and resources allocated to SaaS components.

1.6 Research Contribution

This thesis formulates Software-as-a-Service placement problem using service level agreement constraints and resource constraints. In this thesis, Genetic Algorithmic framework for Software-as-a-Service placement on virtual machines in Cloud computing infrastructure has been proposed. The performance of GA based on Software-as-a-Service placement policy has been compared with First-fit Randomized Algorithm.

1.7 Thesis Layout

This thesis is organized as follows:

- Chapter 2 is having detail background information needed for the research in subsequent chapters. This includes Cloud Computing, Software as a Service, and Genetic Algorithms, Current State of Art for SaaS Placement, and Problem Formulation.
- Chapter 3 is having detail about the proposed Framework. This chapter includes information about the proposed GA framework (like input, fitness function, encoding schema, and GA operations) and contains the different experimental results.
- Chapter 4 gives the detailed conclusion of research work and also shows the future work.

Chapter 2

Cloud Infrastructure for SaaS Placement

2.1 Introduction

In present cloud computing is becoming more popular among IT service providers. The term cloud was coined by CEO of Google Eric Schmidt, used the term cloud to describe the Google service [1]. A study held by IDC identified cloud computing as one of the predominant technology trends in present time [6]. A research conducted by United Kingdom indicates that there are highly interested organizations in UK, which use cloud computing services [12].

2.2 Cloud Computing

Buyya et al. [7] Cloud computing is a parallel and distributed computing system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level-agreements (SLA) established through recognition between the service provider and consumers. Vaquero et al. [53] had stated that clouds are a large pool of easily usable and accessible virtualized resources. These can be dynamically configured to adjust to a scalable load, allowing also for an optimum resource utilization. In cloud pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the infrastructure provider by means of customized SLAs. McKinsey and Co. Report [21] claims that clouds are

hardware-based services offering compute, network, and storage capacity where; hardware management is highly abstracted from the buyers, buyers incur infrastructure costs as variable OPEX, and infrastructure capacity is highly elastic. Cloud computing enables convenient, on-demand access to a shared group of configurable computing resources (networks, servers, storage, applications, and services) that can be rapidly placed and used with minimal managerial effort or service provider interaction, on to the broadband network [45]. A report from the university of California Barkeley [22] summarized the characteristics of cloud computing as:

- the illusion of infinite computing resources.
- the elimination of an up-front commitment by cloud users.
- the ability to pay for use.

Based on the above definitions, in this research, cloud computing will be referred as:

A pool of computing infrastructure to provide pay-per-use services to the end users over a broadband network. The cloud's business model is based on an on-demand model or on the subscription for a limited period of time. All the services are provided by service providers to the users as per SLA norms.

2.2.1 Cloud Architecture

As per NIST given Cloud Computing Reference Architecture (CCRA), which is presented in the following figure 2.1. CCRA identifies the major actors such as Cloud Consumer, Cloud Service Provider, Cloud Auditor, Cloud Broker, and Cloud Carrier, and their functions in cloud computing. NIST CCRA is well suited for describing services, business, or operational relations [5].

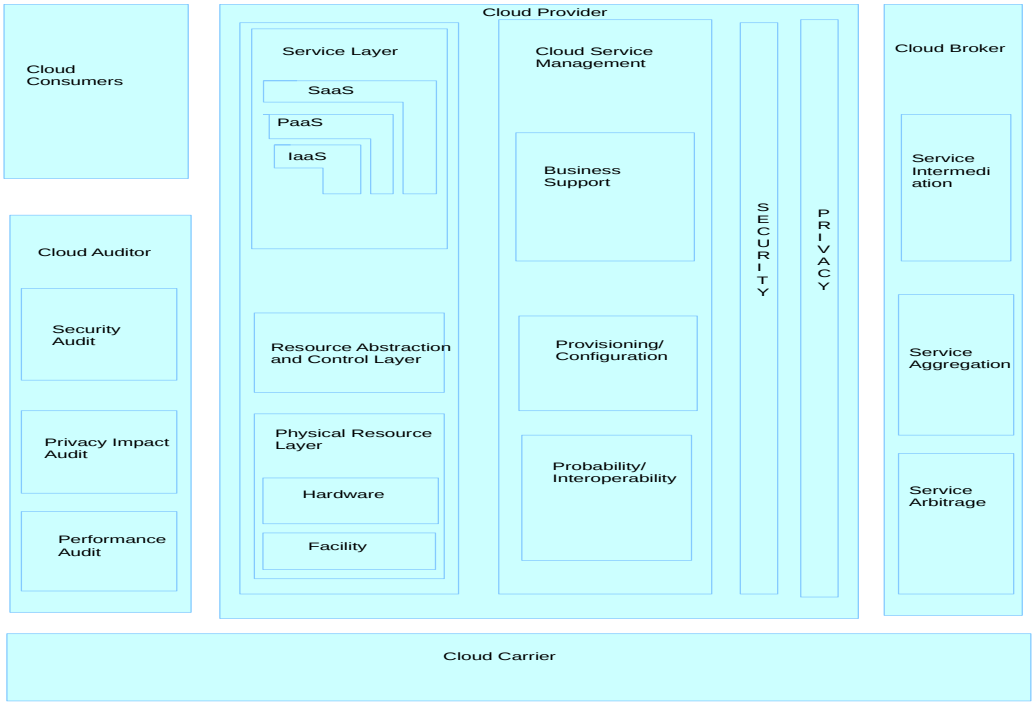

Figure 2.1: NIST Cloud Computing Reference Architecture (CCRA) [5]

2.2.2 Cloud Service Models

NIST divides cloud services in five different categories: IaaS, SaaS, PaaS, NaaS, and CaaS [26]. Most of the researchers categorized cloud services into three main categories: IaaS, PaaS, and SaaS. These three forms of cloud services are three pillars on top of which cloud solutions are provided to the end users. IaaS is a cloud model, which allows users to use computing resources for computation, storage, and networking. For example Amazon EC2 and S3. PaaS is a cloud model which provides cloud resources and proper software platform to develop, deploy, and manage the execution of applications. For example Google App Engine, Microsoft Azure etc. SaaS is a cloud model which refers to browser-initiated applications over thousands of cloud customers. For example Google Gmail and Docs, Microsoft SharePoint etc [29].The relationship between these three cloud services

is defined by cloud computing stack. Data center or physical plant/building is the lowest layer of the cloud computing stack and hosted application or suites of services is the top most layer. In between them, deployment tools and data management, operating systems, virtualization, servers and storage, and network firewalls/security are sequentially defined from top to bottom [3].

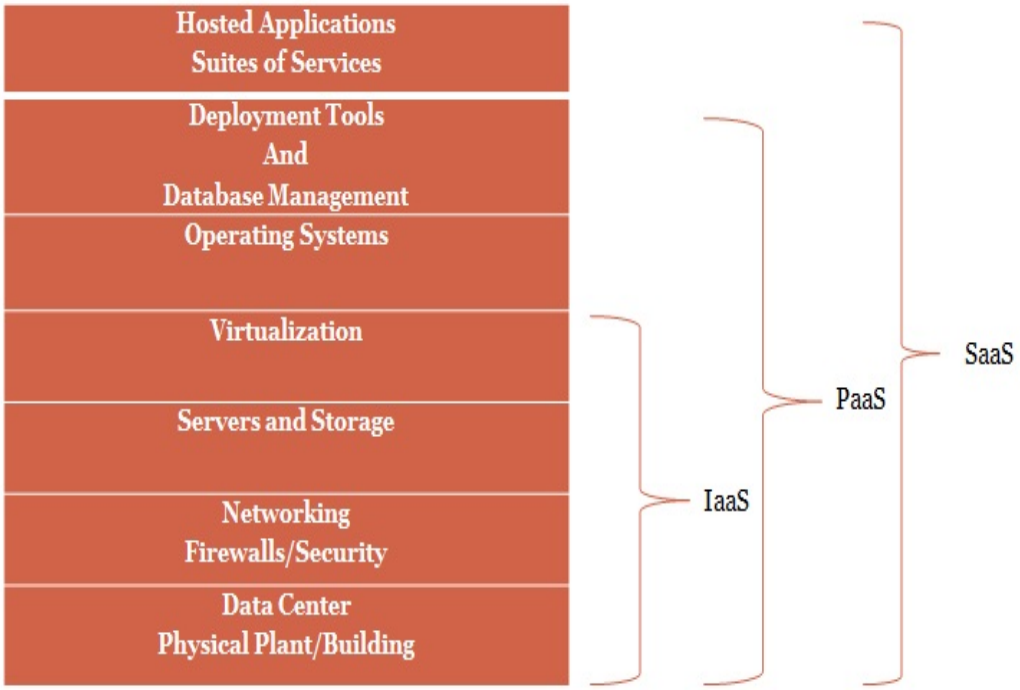

Figure 2.2: Cloud Stack

2.2.3 Cloud Deployment Models

NIST categorized the cloud computing deployment model into four categories: private, community, public, and hybrid. 1) Private cloud: it is a cloud infrastructure, which is operated completely for a single organization. Private cloud may be managed by the same organization or a third party and may be located premise or off premise. The reason behind to set up a private cloud within an organization is having several aspects like: maximize and optimize the utilization of resources,

security concerns, etc. 2) Community cloud: it is a cloud infrastructure, which is shared by two or more organizations and supports a specific community that has shared concerns, policies, requirements, and values. It may be managed by several organizations or a third party and may be located premise or off premise. Community cloud develops for democratic equilibrium and economic scalability. 3) Public cloud: it is a cloud infrastructure, which is made available to the general public or a large industry group. Public cloud is owned by an organization for providing the cloud services. 4) Hybrid cloud: it is a cloud infrastructure, which is a composition of two or more private, community, or public clouds that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability, for example cloud bursting for load-balancing between clouds [2] [19] [55].

2.3 Cloud Infrastructure Model

Cloud infrastructure basically stands for the cloud data center. Data centers are the basic of cloud computing infrastructure, which provide hardware resources to cloud computing services. Data center consists of thousands of servers and established in less dense areas with minimum energy rates and lower chances of natural disasters [50]. Each server has its own processing, input-output, memory, and storage capacity [46]. Google Inc., Microsoft and Amazon are the examples of large data centers to support the cloud services [56]. Cloud computing data centers consist of computation servers and storage servers. These servers are interconnected via physical connections. Each server has one or more VMs installed on it. VMs are slices of resource capacities of servers.

2.4 Cloud software as a service (SaaS)

SaaS came into the picture before the cloud computing. Initially SaaS had been placed onto the SaaS vendor's physical servers or resources and services provided to the users via web [32]. With the increasing demand of SaaS, SaaS vendors tried to find out the solution to these growing demands and they found a solution

for this problem to place the SaaS in cloud computing infrastructure because it provides scalability [35]. NIST defined SaaS as a category of cloud services where the capability provided to the cloud service user is to use the cloud service providers applications running on a cloud infrastructure. Software as a Service application, mostly consumed by web browsers and some are consumed as web services using other client services like desktop and mobile applications [26]. For examples: Google Apps, Microsoft Office-365 [29]. Another definition of SaaS given by Frederick and Gianpaolo [10], Software as a service are placed as a hosted service and access over the broadband network.

2.4.1 Characteristics of SaaS

SaaS separates the owner of software services and the end users of those services [4] [51]. Before SaaS there were two different approaches: traditional software approach and Application Service Provider approach. In traditional software approach customers bought the software product and install on their own machine. The software comes in CD installation and its manual package, and its cost includes maintenance cost by the vendor [56]. ASP is an approach in which, software still bought by a customer and install at the ASP data centers [34]. In case of SaaS, the software placed on the cloud vendor's servers and clients use this software on pay-per-use basis via internet [42].

SaaS is a business model. In conventional software, providers offer software to users on the one-time pay basis. This cost includes the license cost to use that software as well as maintenance cost. In addition, user have to bear the hardware and its maintenance cost. ASP decreased some of the users cost, by deploying their software into data centers. In this user charge for software license, hosting and maintenance [34]. But in case of SaaS user are charged pay-per-use basis. Users do not have to require to purchase the complete software license as in ASP and traditional software approach. In SaaS the ownership of hardware and software shifts from users to the providers. The infrastructure cost is shared by several users in place of single user in ASP or traditional approach. Due to a large number of users of the same software the cost becomes too small [56].

Multi-tenancy concept of SaaS design separates it from ASP or traditional software approach. Multi-tenancy is the concept, by exploiting of this, hardware and software instances are shared between users, hence multiple end users can utilize the same software and hardware instances. It helps to lowering the cost of providing software services [46].

2.4.2 SaaS Examples

There are so many companies which offer Software as a Service. In this section we will discuss about a few of them like Microsoft, Google, Salesforce, and IBM software as a service [18] [54]. A brief information is given in following paragraphs.

Microsoft offers Microsoft Office Live Small Business [16]. Microsoft Office Live Small Business offers features like storage manager, an e-commerce tool to help small business to sell products, and E-mail Marketing Beta, to send emails [16]. Microsoft also offers Office 365 for home (Rs. 420.0 per month), personal (Rs. 330.0 per month), and business [17].

Google offers Google Apps for Business. Google Apps includes services for collaboration and communication designed for all the business of different size. Google Apps for Business is available for US$10 per-user-per-month with Vault US$5 per-user-per-month without Vault. Google offers Gmail, Docs, Drive, Hangouts, Sites, Vaults, and Spreadsheets etc. as SaaS [13].

Salesforce.com offers CRM services. Salseforce.com charge US$5 per-user-per-month for Contract Manager, US$25 per-user-per-month for Group, US$65 per-user-per-month for Professional, US$125 per-user-per-month for Enterprise, and US$300 per-user-per-month for Performance [33].

IBM offers a SaaS solution under the name "Blue Cloud". Blue Cloud will allow corporate data centers to operate by enabling computing across a distributed and globally accessible resources. It is based on open-standard and open-source software supported by IBM [14].

2.4.3 SaaS Component Model

SaaS is delivered in composite part, in which each and every component are loosely coupled in nature and communicate with each other to provide a high-level functional system to the users [30]. There are two maturity models, proposed by Microsoft [28] and proposed by Kitagawa et al. [39]. These two maturity models indicate that SaaS placed in a cloud computing environment with features like scalability, configurability, elasticity, and multi-tenancy. Hence the complete functionalities of a service are achieved by composing one or more software components in order to develop a SaaS with high-level functionalities [56]. Hence SaaS is the composition of several software components like; application component (AC), integration component (IC), business component (BC), and storage component (SC) [56] [59]. The components are distributed on physical servers due to certain constraints; like data components need to be placed on storage servers and application component needs to be placed on computational servers due to their on basic requirements. All the components are placed on top of the VM in cloud computing infrastructure which are deployed on physical or storage servers. For efficient placement of SaaS it is required that SaaS don't violates resource constraints [58] as well as SLA constraints [43].

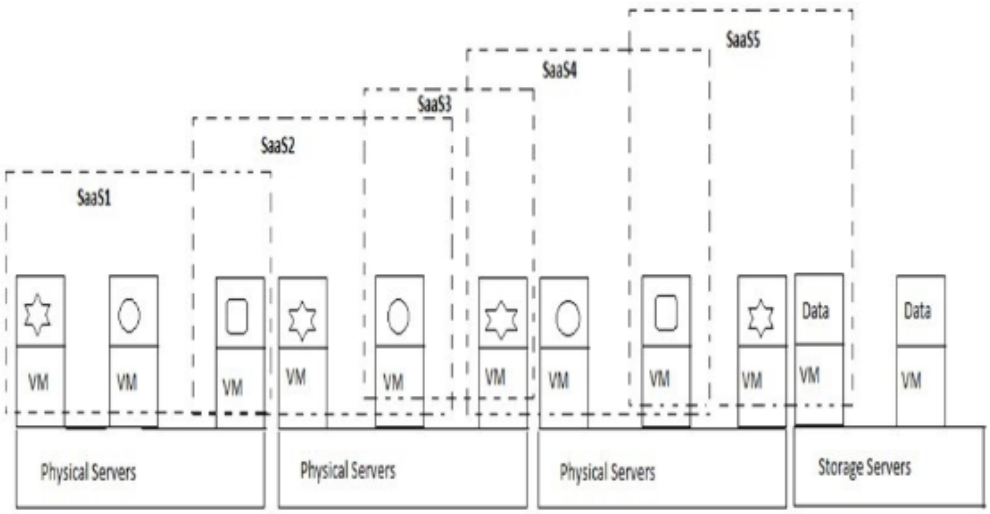

Figure 2.3: SaaS component model in cloud computing infrastructure [58]

2.5 Service Level Agreement

Service level agreement (SLA) is a contract between end users and service provider. SLA is having information about all service requirements, that are formally specified service performance and corresponding revenue cost [43]. In case of cloud computing trust between customer and provider also comes into consideration, mainly for enterprise customers that may outsource its critical data. SLA serves as the base for the expected level of agreed services between the consumer and the provider. The QoS attributes are generally part of an SLA; response time and throughput. Due to the QoS parameters change constantly and to enforce the agreement, these parameters required to take care [48].

2.6 SaaS deployment constraints

SaaS deployment constraints adopted from [43] [57], which are as follows:

2.6.1 Resource constraints:

The total resource requirements for SaaS components that are placed in either compute servers/storage servers or virtual machines must not exceed the VM's resource capacity.

2.6.2 Placement Constraints:

Two types of placement constraints present:

a An anti-location constraint: this determines a set of VMs that should not host a specific SaaS component.

b An anti-collocation constraint: this determines a set of components that cannot be deployed on the same VM.

2.6.3 Execution time constraints:

To ensure the optimal performance of the SaaS, the placement of the SaaS components is based on its estimated total execution time. For the SaaS resource

optimization problem, the execution time is considered as users SLA and given as an input. The total execution time is calculated based on four numerical attributes which are:

a. the time taken for transferring data between the storage servers and the virtual machine,

b. the processing time of a component in a selected virtual machine,

c. the execution time of a path in the SaaS workflow, and

d. the sum of the execution time of the critical path of each workflow multiplied by its weighting.

Based on these four values, the total execution time of the SaaS is determined. Total execution time must not exceed the maximum response time of a SaaS as agreed in the user's SLA.

$$total\ execution\ time \leq maximum\ response\ time$$

2.6.4 Sequence of migration constraints:

To change the placement from one VM to another, the solution has to be considered the sequence SaaS components that need to be replaced based on their current placement. There are two scenarios consider:

1. Sequential move: A particular component can only be moved when another one has been completed. This is in the case of where two component's migrations cannot be done in parallel because the destination VM contains another component that's due to be migrated. As such, the latter component needs to be moved first to free some resources for the other component.

2. Cyclic move: A set of component's migration may need an intermediate destination machine.

This is in the case of when two or more components need to be exchanged places. This can create a cyclic constraint if the machines involved have insufficient resources.

2.6.5 Cost Constraints

The cost efficient VM combination consists, k number of instances. Each and every VM cost depends upon the resource on which they are deployed. A VM can be chosen only if, profit of the company maximizes. That means expected revenue must be greater than the expected cost.

$$Max\ profit = expected\ revenue\ -\ expected\ cost$$

For choosing any particular VM, to deploy a SaaS component Max profit must be greater than zero.

2.7 Problem Formulation

In a Cloud infrastructure a set of servers is connected via communication network links. The computing availability of these servers is made available to users through virtual machines. Virtual machines are placed onto the servers, which have their own resource capacities; memory, processing, input-output, and storage capacities. There are a set of SaaS components, which needs to be placed onto the VMs and have their own requirements; memory, processing, input-output, and storage requirements. The objective is to find out an a optimal set of VMs on which SaaS components can be deployed and give maximum profit to the service provider as well as reduce the charges of customer charged by the service provider. This optimal set of VMs must satisfy the resource constraints as well as SLA constraints.

The cloud infrastructure, SaaS component, objective, and constraint formulation are given below:

Cloud Infrastructure, includes cloud data center that means pool of physical servers $PS = \left\{ps_1, ps_2, ps_3, ..., ps_r\right\}$ and set of virtual machines (VMs) $VM = \left\{vm_1, vm_2, vm_3, ..., vm_n\right\}$, deployed on those physical servers. The resource capacities of VMs are represented in tuple form (PC_{vm_i}, MC_{vm_i}, ST_{vm_i}, IOC_{vm_i}), $1 \leq i \leq n$. Where PC_{vm_i} is processing capacity of vm_i, MC_{vm_i} is main memory capacity of vm_i, ST_{vm_i} is the storage capacity of vm_i, and IOC_{vm_i} is the input-

output capacity of vm_i.The cloud modelling presents a general information about VMs and their resource capacities.

Table 2.1: Sets and Attributes of Physical Resources in Cloud

Resources	**Description**
$ps_i \in PS$	The i^{th} physical server ps_i in PS, where PS is a set of physical server $i \leq r$
$vm_x \in VM$	The x^{th} virtual machine vm_x in VM, where VM is a set of virtual machines $x \leq n$
PC_{vm_x}	The processing capacity of x_{th} virtual machine
MC_{vm_x}	The memory capacity of x_{th} virtual machine
ST_{vm_x}	The storage capacity of x_{th} virtual machine
IOC_{vm_x}	The Input-output capacity of x_{th} virtual machine

SaaS Component, $SC = \left\{sc_1, sc_2, sc_3, ..., sc_m\right\}$ are fixed. Each and every SaaS component is placed on a single virtual machine. The resource requirements of SaaS Components are represented in tuple form (TS_{sc_i}, M_{sc_i}, SZ_{sc_i}, IOC_{sc_i}), $1 \leq i \leq m$ and $m << n$. Where TS_{sc_i} is task size of sc_i, M_{sc_i} is a main memory requirement of sc_i, SZ_{sc_i} is size of sc_i, and IOC_{sc_i} is an input-output requirement of vm_i. The SaaS modelling presents a general information about software components and their resource requirements.

Table 2.2: Sets, Parameters and Requirements of Software as a Service

Resources	**Description**
$sc_i \in SC$	The i^{th} SaaS component sc_i in SC , where SC is a set of SaaS component $i \leq m$
mrt_{sc_i}	The maximum response time of sc_i
TS_{sc_i}	Task size of sc_i
M_{sc_i}	Memory requirement of sc_i
SZ_{sc_i}	Size of sc_i
IOC_{sc_i}	Input-output requirement for sc_i
TET_{sc_i}	Total execution time for sc_i
mrt_{sc_i}	Maximum response time for sc_i defined in SLA

Objective: To maximize the profit of SaaS provider via finding the optimum placement of SaaS components.

$$Maximize \Delta Pr = \sum_{j=0}^{m} RS_{sc_j} - \sum_{i=1}^{n} \sum_{j=1}^{m} C_{vm_{i,j}}$$

where ΔPr is the profit of the SaaS provider, RS_{sc_j} is the revenue cost of j^{th} SaaS component and $C_{vm_{i,j}}$ is the cost of resource when j^{th} component placed on i^{th} virtual machine.

$$C_{vm_{i,j}} = o_{i,j} * c_i * ta_i$$

Where $o_{i,j} = 1$, iff j^{th} component deployed on i^{th} virtual machine, otherwise $o_{i,j} = 0$. c_i is the rental cost of i^{th} virtual machine, and ta_i is the time for which i^{th} virtual machine acquired by the SaaS.

$$RS_{sc_j} = rc_j * t_j$$

Where rc_j is the revenue cost of j^{th} SaaS component per unit time and t_j is the service time of j^{th} SaaS component.

Constraints: The placement of SaaS components over VM is depending on VM's capacities. We can place a SaaS component over a virtual machine if and only if the requirements of the SaaS component must be less or equal to that particular virtual machine. Following equations represents the SaaS component constraints which include resource and SLA constraints.

$\forall vm_x \in VM \sum_{sc_i \in SC} TS_{sc_i} \leq PC_{vm_x} \mid P(sc_i) = vm_x$

$\forall vm_x \in VM \sum_{sc_i \in SC} M_{sc_i} \leq MC_{vm_x} \mid P(sc_i) = vm_x$

$\forall vm_x \in VM \sum_{sc_i \in SC} SZ_{sc_i} \leq ST_{vm_x} \mid P(sc_i) = vm_x$

$\forall vm_x \in VM \sum_{sc_i \in SC} IOC_{sc_i} \leq IOC_{vm_x} \mid P(sc_i) = vm_x$

$\forall sc_i \in SC \; TET_{sc_i} \leq mrt_{sc_i}$

2.8 Current State of Art of CPP

Component placement problem (CPP) is divided into two categories: 1) online CPP and 2) offline CPP. The following table illustrates the technique used for SaaS placement. Our approach of placement is justified by the prior techniques used for placement purpose.

Table 2.3: State of Art of SaaS Placement

Researches	**Technique used**
A Karve et al. [37]	Proposed middleware clustering technique capable to dynamic allocation of web services
Zhu et al. [61]	Address the placement problem for application components with storage components
Zimmerova et al. [62]	Proposed a placement technique focused on communication between components and communication was captured by automata language
Kichkaylo et al. [39]	Proposed placement technique focused communication between components and application defined by components
Urgaonkar et al. [52]	Used first-fit approximation algorithm for offline CPP
Thomas et al. [40]	Proposed placement technique which maximizes cost saving and minimizes resources, and calculate resource requirements of multi-tenant application instances
Zorath Izzah Mohd et al. [57]	Used GA to find the placement of components with resource constraints
Moens et al. [46]	Define feature-based cloud resource management model for this they had used SPLE and SOA. They used meta heuristic algorithms for placement of SaaS
Zhipiao Lia et al. [43]	Used GA for cost-aware placement of SaaS

2.9 Summary

The objective of the thesis is to formulate CPP and design algorithm for CPP. This chapter presented the background of research, cloud modelling, and problem formulation. The current state of art for CPP, proved that the CPP is a candidate for genetic algorithm and also gave the idea about different techniques used for

CPP. The problem formulation showed the cloud modelling, objective function, and different constraints (resource and SLA), which must not violate.

Chapter 3

GA Framework for SaaS Component Placement

3.1 Introduction

SaaS offers flexible and scalable services to the end users or clients. SaaS can be expanded or shrink according to the user's requirements. In each and every situation, the customization of SaaS can be performed by the end users at client side or by the service provider at the front end or server side. Two examples of such scenario are Google Apps offered by Google [13], and Microsoft Office Live offered by Microsoft [17]. Microsoft offers two different categories of the Microsoft Office Live, based on the functionalities of its own Microsoft Office Live, where one is for home users and another is for business users. Microsoft Office Live for Home is further divided the services into three sub-categories Office 365 Home, Office 365 Personal, and Office Online. Microsoft Office 365 provides different services like Word, Excel, PowerPoint, Outlook, OneNote, and Lync. Google offers two different categories of the Google Apps, based on the functionalities of its own Google Apps, where one is Google Apps for Business without Vault and another is Google Apps for business with Vault. Google Apps provides different services like Gmail, Calender, Drive, Docs, Hangouts, Vault, and more other services. The flexibility and scalability of SaaS are also referred as elasticity of software as a service. In our research, SaaS is in composite form. The parts of the SaaS referred as SaaS components, can be placed on its own. Each SaaS component represents a unique well-defined function of the software. These components can

be application component (AC), integration component (IC), business component (BC), and storage component (SC) [56] [59].

The full utilization of SaaS components bring a number of advantages to, SaaS provider and SaaS users. These advantages include, 1) reduce resource cost, 2) increase profit of SaaS provider, 3) flexible and scalable, offers of SaaS services, and 4) reduce the subscription cost of end users. Managing such SaaS components raises few new challenges. Initial or offline placement of SaaS components in one of the challenges from them, which finds the appropriate VMs for the deployment of SaaS components. The problem of SaaS component placement on top of the virtual machines which are deployed on Cloud servers, is referred to as Component Placement Problem (CPP). The aim of finding the solution of placement problem is to find-out which VM can beer which component, such that the resource and SLA constraints should not violated and it provides flexible services to the users. The servers on which VMs are deployed, located across the world in a particular Cloud network.

Component placement problem concerns with finding the optimum set of VMs on which SaaS components can be placed such that all user requirements should be satisfied and give maximum profit to the SaaS providers [39] [43] [52] [58] [61].In CPP, the solution of the problem is subject to a set of resource and SLA constraints. Although extensive research has been carried out for component placement problem, their solution does not take account of resource constraints and SLA constraints both together. This research presents a placement algorithm that considers both SLA constraints as well as resource constraints and gives a cost-aware solution for component placement. The CPP has been proven by researchers as an NP-complete [36] [37] [40] [44] [56], the proposed GA gives a sub-optimum solution for the problem.

3.2 Genetic Algorithms

Finding an optimal solution to the large scale, complex combinatorial problems is not a practicable option, due to its vast amount of computing time needed

to find such solutions. In practical a good solution, obtaining at the reasonably small computing time by a heuristic, is often the only possibility [49]. It had been proved that Genetic Algorithm (GA) is suitable to solve such large scale complex combinatorial problems [9]. Genetic Algorithms were introduced by Fraser and after that developed by Holland [27]. GA is a stochastic process of searching, which imitates the process of biological evolution, especially in the selection process as well as recombination operation [24]. GA finds the best solution manipulating a set of candidate solutions in which the fittest solution is having higher chance to survive and the solutions combine with other solutions to get the new solutions into that population. GA are most suitable for a large scale, complex search optimization problem as compared to other search algorithms like enumerative or random search algorithms [56]. Enumerative search technique considers most of the solutions in the solution space, but it tries to reduce the size of the solution space by applying some heuristics [41]. Hence it is suitable only if the solution space is small. GA has a random element (selection procedure) in it, but the search is directed by the environment. Algorithm starts with a set of solutions (represented by chromosomes) called a population. In GA solutions are taken from the parent population and used to generate new or child population. This is imposed by a wish, that the child population will give a better solution than the parent population. From the parent solutions populations are selected to produce child solutions (offspring) are selected on the behalf of their fitness values. That means, there is more chances of selecting the most suitable solution [47].

3.2.1 GA Parameters

The ability of Genetic Algorithms to find-out sub-optimal solution mainly depends on its implementation and operations. The GA has eight parameters, which are 1) the genetic representation of candidate solutions, 2) the population size, 3) the evaluation function, 4) the genetic operators, 5) the selection algorithm, 6) the generation gap, 7) the amount of elitism used, and 8) the number of duplicates allowed [47].

- **The genetic representation of candidate solutions**, also called population and represented by chromosomes.
- **The population size**, says how many chromosomes are in the population.
- **The evaluation function**, describe that the fitness of the solution.
- **The genetic operators**, used to generate new population. Genetic operators are crossover and mutation operators, which are based on the corresponding probabilities.
- **The selection algorithm**, describe how to select these chromosomes. There are few methods for selection procedure as the roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.
- **The generation gap**, describes the number of iterations.
- **The amount of elitism used**, to increase the performance of GA, because it prevents a loss of the best found solution by copying the best chromosome to the child population from the parent solution.
- **The number of duplicates allowed**, used to copy the best solutions.

3.3 SCPGA

In chapter 2, the size of Cloud's infrastructure is discussed. There are large scale resources available in Cloud's infrastructure. Hence Component Placement Problem can be considered as a large-scale complex combinatorial problem that deals with finding of VMs for component placement with a set of constraints should be followed. The goal has optimized the resources and maximize the profit of SaaS providers.

The SaaS Component Placement based on Genetic Algorithms (SCPGA), is a search heuristic inspired from the theory of Biological evolution. This technique is based on population that represents the set of candidate solutions of the problem,

and evolves to generate new candidate solution through GA operations for finding the optimal solution.

The SCPGA is developed for CPP. The population of SCPGA consists a set of candidate solutions which represents the placement of SaaS components on top of the virtual machines. After a number of generations the optimal solution is achieved. In these numbers of generations, crossover, mutation, and selection operations are applied in the algorithm. Algorithm 1 describes the SCPGA.

Algorithm 1 SCPGA

Data: Initial Population
Result: Sub-Optimal solution for SaaS Placement
Initialize bestFitness
Random initialize(Population)
while *the termination condition is not true* **do**
  for $P \in Population$ **do**
    if *P violates the requirement constraints and SLA constraints* **then**
      Repair(P)

    end
    Calculate rental cost of VMs
    Calculate the revenue cost
    Calculate the profit of SaaS provider
    if $\Delta Pr > bestFitness$ **then**
      Replace bestFitness and store P

    end
  end
  Select individuals from Population via roulette wheel selection
  Probabilistically apply single point crossover operator to generate new individuals
  Probabilistically select individuals and apply mutation operator to generate new individuals
  Replace the individuals of old population by new individuals

end
output bestFitness and best individuals

3.3.1 SCPGA Encoding

A chromosomes in the SCPGA represents the placement for the SaaS components. The chromosomes contains m number of genes, each of which corresponds to the

SaaS component, representing the ID of virtual machines where the SaaS components should be placed and where m is the number of SaaS components. Figure 3.1 shows the encoding schema of the SCPGA.

SaaS Components	SC_1	SC_2	SC_3	SC_4	SC_5	SC_6	SC_7	SC_8	SC_9	SC_{10}	.....	SC_{m-1}	SC_m
Virtual Machines	42	1032	234	88	732	70	114	1	97	4	.....	6	423

Figure 3.1: Encoding schema of the SCPGA

3.3.2 Infeasible Encoding Problem

The representation naturally maps CPP into a chromosome of the SCPGA. The individuals of chromosome generated randomly in the initial population, and the genes generated may not be feasible to the placement. For example the VM's memory capacity is 4GB and the components memory requirement is 5GB, the genetic operator may produce that VM's ID for that particular component, but it is not possible to place that particular component on the generated VM. To handle this infeasible encoding problem, a repairing technique is used. The repairing technique performs checks on each gene to find any infeasible individuals. If any gene is found infeasible, another random number is generated based on the correct search space.

3.3.3 Genetic Operators

There are three basic genetic operators involve in generating the optimal solution. These operators are described in the following sections.

Selection

The selection is roulette wheel selection. The selection process is stochastic selection from the current generation to create the parents for the next generation. The requirement of the selection process is to choose fittest individuals because fittest individuals have a greater chance of survival. Hence, weaker individuals

will have less chance to choose as parents. This process of selection is known as roulette wheel selection.

Crossover

The crossover operation is single point crossover. The crossover operation depends on the crossover probability, which gives the information about number of chromosomes would be selected for the crossover operation. The point of crossover is in between the segments of individuals in chromosome and it would be generated randomly. The crossover operations exchange the sub-string from two selected parents and generates two children. Figure 3.2 illustrates the crossover operation.

Before Crossover:

sc_1	sc_2	sc_3	sc_4	sc_5	sc_6	sc_7	sc_8	.....	sc_{m-3}	sc_{m-2}	sc_{m-1}	sc_m
42	1032	244	88	732	80	98	103	.....	15	6	9	423

sc_1	sc_2	sc_3	sc_4	sc_5	sc_6	sc_7	sc_8	.....	sc_{m-3}	sc_{m-2}	sc_{m-1}	sc_m
54	97	72	1024	244	4	12	16	.....	21	32	13	449

After Crossover:

sc_1	sc_2	sc_3	sc_4	sc_5	sc_6	sc_7	sc_8	.....	sc_{m-3}	sc_{m-2}	sc_{m-1}	sc_m
42	1032	244	88	732	80	98	103	.....	21	32	13	449

sc_1	sc_2	sc_3	sc_4	sc_5	sc_6	sc_7	sc_8	.....	sc_{m-3}	sc_{m-2}	sc_{m-1}	sc_m
54	97	72	1024	244	4	12	16	.....	15	6	9	423

Figure 3.2: Crossover operation in SCPGA

Mutation

The mutation operation is knowledge-based, which changes the VM to a particular SaaS component with a new VM, such that the new VM is more appropriate for the placement of that particular component. This new VM reduces the overall resource cost and maximize the profit of SaaS providers.

The mutation operation is applied on selected genes from the chromosome. This selection is based on the mutation probability. On the basis of mutation probability the number of genes from chromosome are selected for mutation operation. Figure 3.3 illustrates the mutation operation.

Before Mutation:

sc_1	sc_2	sc_3	sc_4	sc_5	sc_6	sc_7	sc_8	.....	sc_{n-3}	sc_{n-2}	sc_{n-1}	sc_n
42	1032	244	88	732	80	98	103	.....	15	6	9	423

After Mutation:

sc_1	sc_2	sc_3	sc_4	sc_5	sc_6	sc_7	sc_8	.....	sc_{n-3}	sc_{n-2}	sc_{n-1}	sc_n
42	1032	244	88	732	80	23	103	.....	15	6	9	423

Figure 3.3: Mutation operation in SCPGA

3.3.4 Decision of Stopping Criteria

This section decides the stopping criteria of the used SCPGA. The following table illustrates the list of parameters used.

Table 3.1: Genetic Algorithmic Parameter for Profit Finding w.r.t. Number of Iteration

Parameters	**Values**
Population size	320
Number of iteration	variable
Selection	Roulette wheel selection
Crossover	Single point crossover
Crossover probability	0.90
Mutation	Single bit mutation
Mutation probability	0.05

The experiment run in a cloud with 16 SaaS components and 400 virtual machines. These values are fixed and the number of iterations varies. Figure 3.4 and Table 3.2 illustrates the profit value of the SCPGA. The figure 3.4 shows the linear characteristics after 300 iterations. Hence, if we further increase the number of iterations the results characteristics will not change.

Table 3.2: Profit finding via GA with fixed number of SaaS components and VMs for deciding the stopping criteria

Number Of Iterations	**Profit via SCPGA** ($*10^6$)
50	6.2132
75	7.6020
100	7.9412
125	7.9284
150	8.3316
175	8.9268
200	8.7348
225	8.9268
250	8.9268
275	8.7252
300	8.9268
325	8.9268
350	8.9268
375	8.9268
400	8.9268
425	8.9268
450	8.9268
475	8.9268
500	8.9268
525	8.9268
550	8.9268
575	8.9268
600	8.9268
625	8.9268
650	8.9268
675	8.9268
700	8.9268

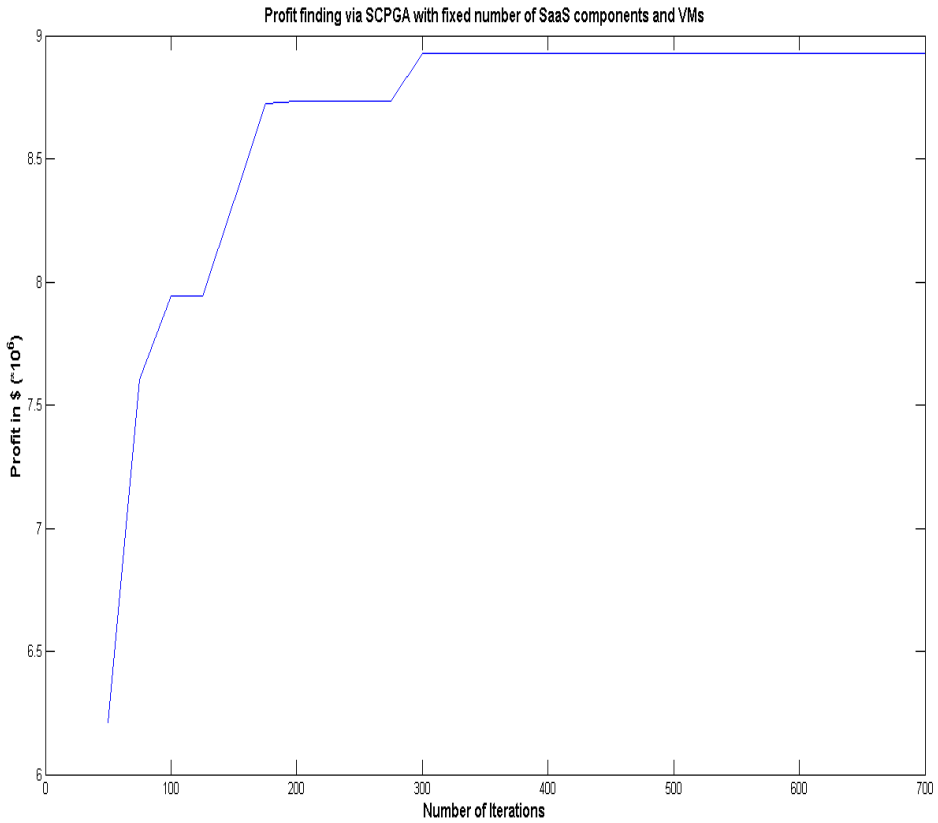

Figure 3.4: Profit finding w.r.t. number of iterations

3.4 Results

The placement problem solving SCPGA has been implemented in JAVA in Eclipse IDE. The experiment evaluates the quality of work. For all experiments, we used the random dataset generated with the help of Amzon's and Azore's datasets [11]. The screen-shots of the dataset are given in figure 3.5 and 3.6. All experiments are carried out in Dell Inspiron 1545 with Ubuntu 13.10 operating system, Intel $Core^{TM}2$ Duo CPU T6600 @ 2.20GHz x 2, and 3GB RAM.

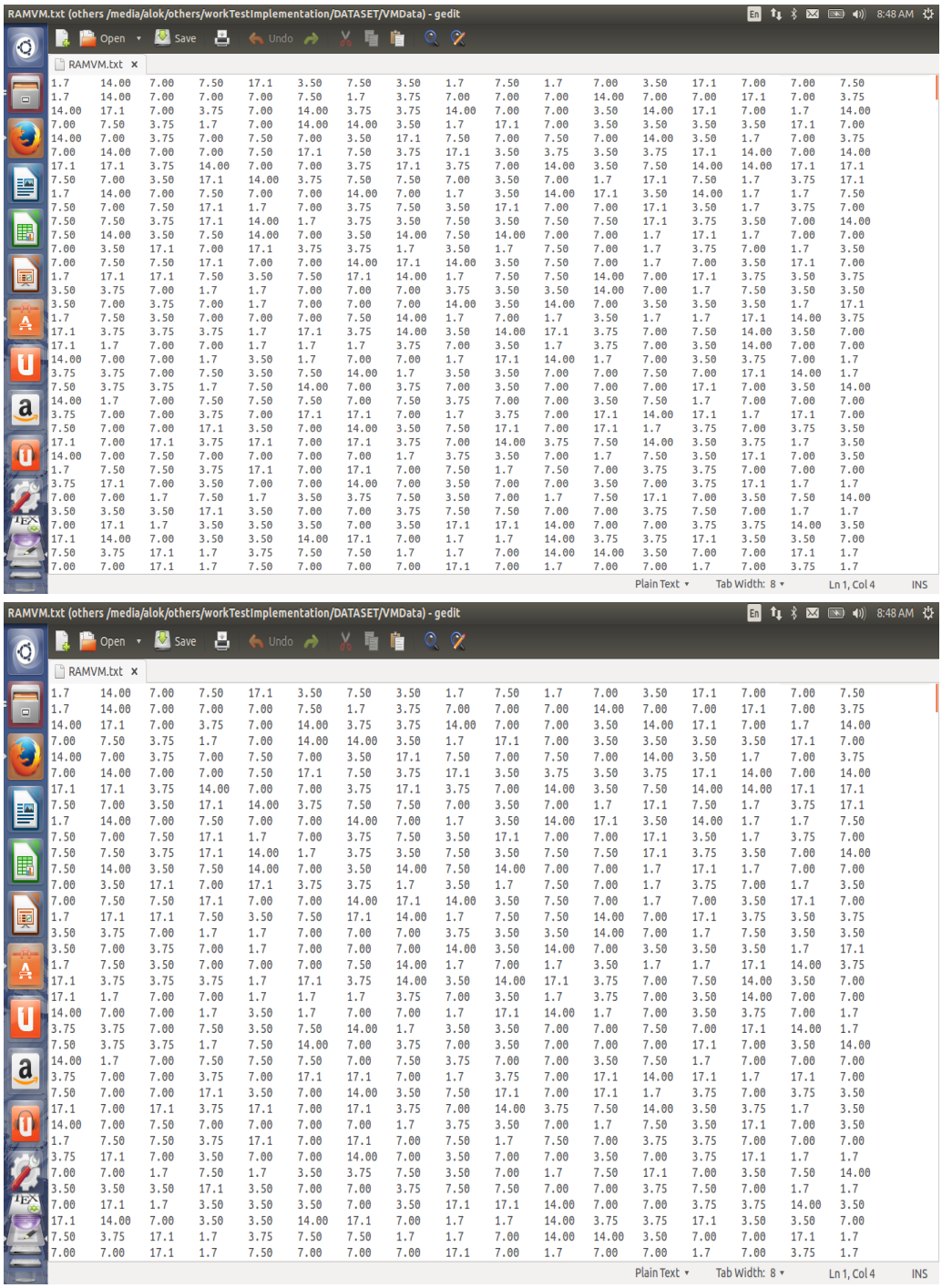

Figure 3.5: Data set screen shot-1

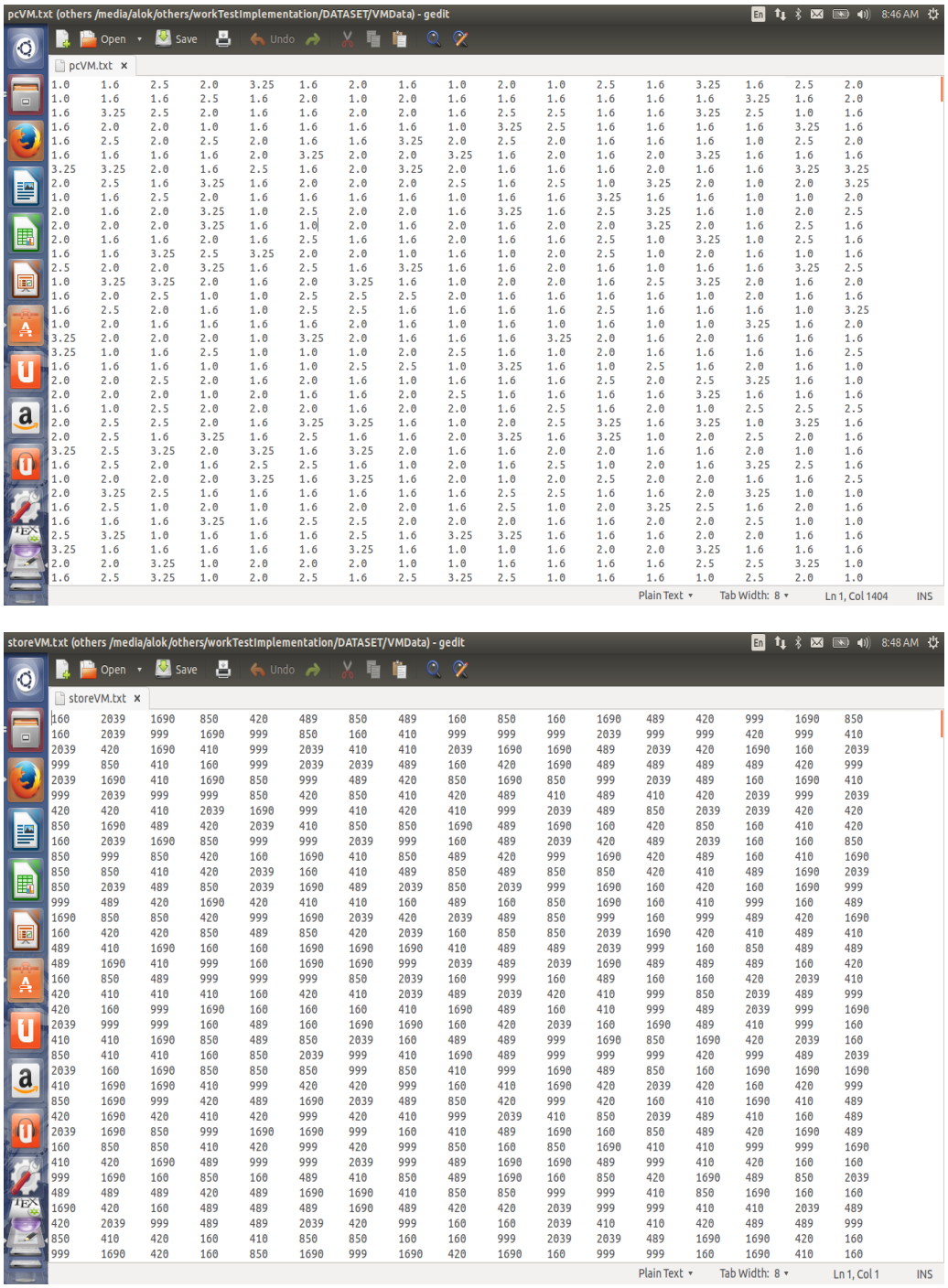

Figure 3.6: Data set screen shot-2

3.4.1 Profit Finding w.r.t. Number of Virtual Machines

The list of parameters used in this experiment are listed in following table.

Table 3.3: Genetic Algorithmic Parameter for Profit Finding w.r.t. Number of Virtual Machines

Parameters	**Values**
Population size	320
Number of iterations	300
Selection	Roulette wheel selection
Crossover	Single point crossover
Crossover probability	0.90
Mutation	Single bit mutation
Mutation probability	0.05

The experiment run in a cloud with fixed number of SaaS components. The number of virtual machines varies. Figure 3.5 and Table 3.4, illustrates the profit value of both SCPGA and First-fit RA. The comparison was based on the calculated profit from objective function. Due to stochastic nature the experiments repeated several times. It can be seen that for all test cases, the SCPGA has always a higher profit value than First-fit RA, which implies that SCPGA provide a better solution for placement of SaaS components.

Table 3.4: Profit finding via SCPGA and Firstfit RA with fixed number of SaaS components

Number Of VMs	**Profit via First-fit RA** $(*10^6)$	**Profit via SCPGA** $(*10^6)$
100	3.3332	8.7252
200	3.1220	8.7060
300	3.1220	8.9268
400	2.8148	8.9268
500	3.3332	8.9268
600	3.3332	8.9268
700	3.3332	8.9268
800	3.3332	8.9268
900	3.1220	8.9268
1000	3.3332	8.9268
1100	3.3332	8.9268
1200	3.3332	8.7252
1300	3.3332	8.9268
1400	3.1220	8.9268
1500	3.0068	8.9268
1600	3.0068	8.9268
1700	3.3332	8.9268
1800	3.3332	8.9268
1900	3.1220	8.9268
2000	3.1220	8.9268
2100	3.1220	8.9268
2200	3.1220	8.9268
2300	3.1220	8.9268
2400	3.3332	8.9268
2500	3.3332	8.9268

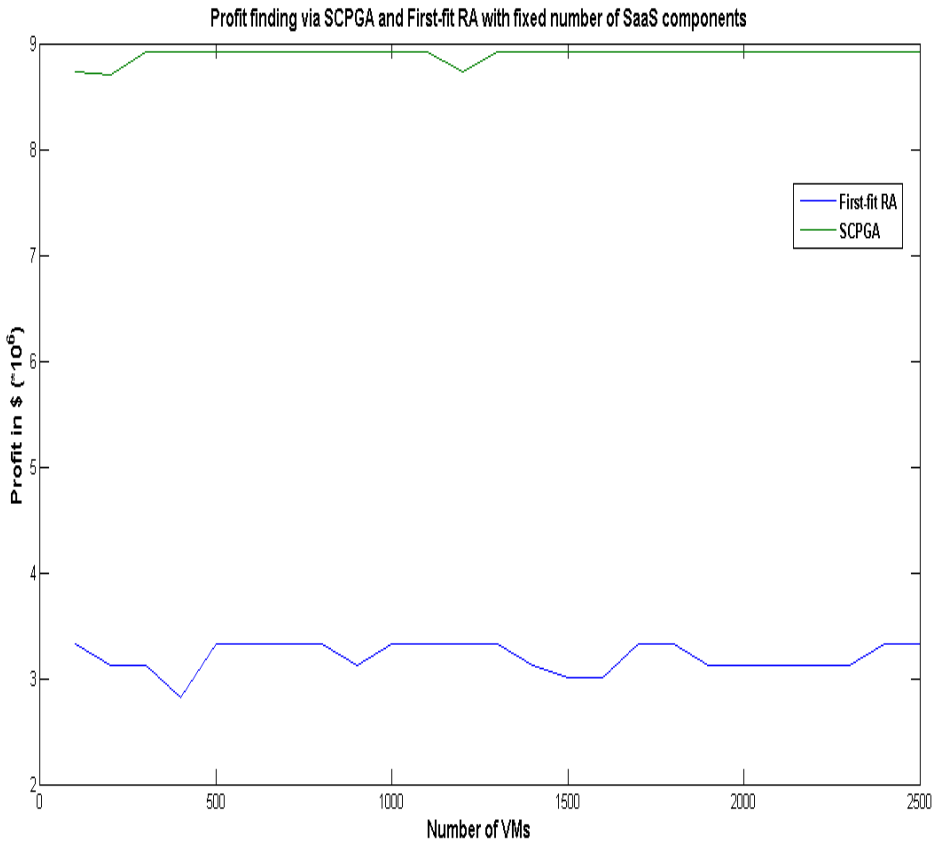

Figure 3.7: Profit finding via SCPGA and Firstfit RA with fixed number of SaaS components

3.4.2 Profit Finding w.r.t. Number of SaaS Components

The list of parameters used in this experiment are listed in following table.

Table 3.5: Genetic Algorithmic Parameter for Profit Finding w.r.t. Number of SaaS Components

Parameters	**Values**
Population size	variable
Number of iteration	300
Selection	Roulette wheel selection
Crossover	Single point crossover
Crossover probability	0.90
Mutation	Single bit mutation
Mutation probability	0.05

The experiment run in a cloud with variable number of SaaS components. The number of virtual machines is fixed. Figure 3.6 and Table 3.6, illustrates the profit values of both SCPGA and First-fit RA. The comparison was based on the calculated profit from objective function. Due to stochastic nature the experiments repeated several times. It can be seen that for all test cases, the SCPGA has always a higher profit value than First-fit RA, which implies that SCPGA provide a better solution for placement of SaaS components.

Table 3.6: Profit finding via SCPGA and Firstfit RA with fixed number of SaaS components

Number Of VMs	**Profit via First-fit RA** ($*10^7$)	**Profit via SCPGA** ($*10^7$)
16	0.33332	0.89268
32	0.45487	1.56655
48	0.66753	2.24569
64	0.88148	2.49508
80	0.81811	2.75571
96	0.91058	3.26570
112	0.9455704	3.8611704
128	1.1424312	4.0763512
144	1.1631512	4.6717112
160	1.2108016	4.9590416
176	1.5041616	5.5524016
192	1.9840608	6.0954208
208	1.7580208	6.5587408
224	1.9950784	7.2075584
240	2.3939984	8.2271984
256	2.5280784	8.6565584
272	2.9087072	9.2313472
288	2.6767552	9.3515552
304	3.143024	9.225024
320	3.496032	10.024432
336	3.2683808	10.0475808
352	3.4972416	10.4504416
368	3.63948	10.49772
384	3.7689088	11.6405888
400	3.9791616	11.5741216

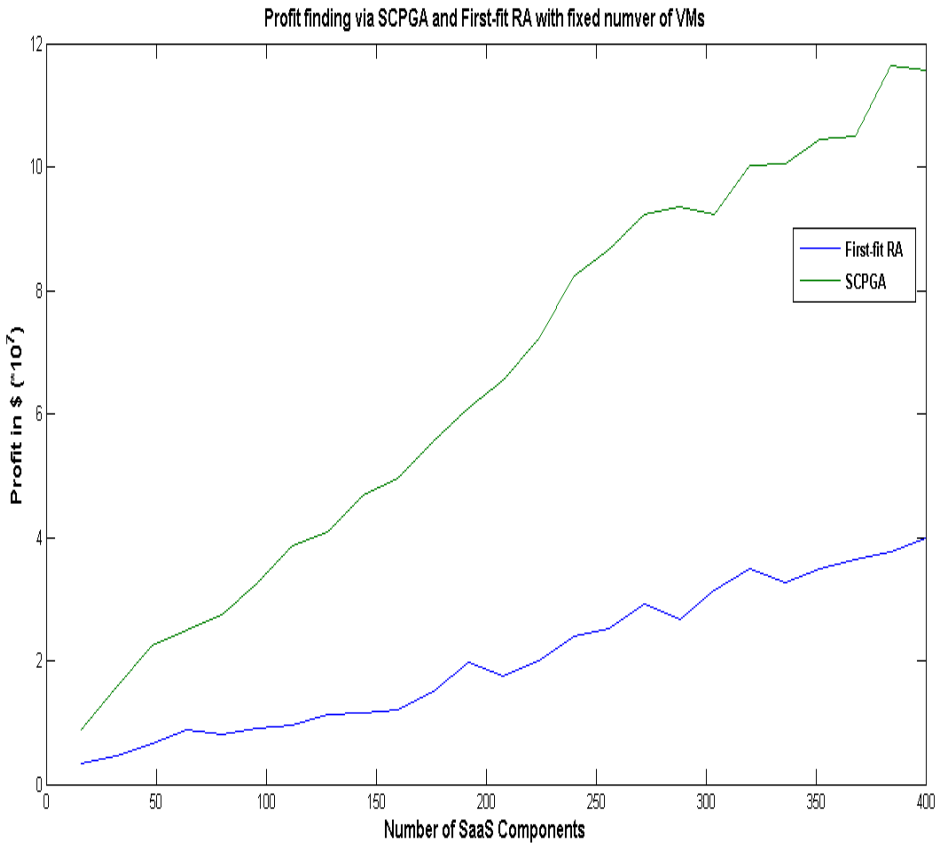

Figure 3.8: Profit finding via SCPGA and Firstfit RA with fixed number of VMs

3.5 Summary

This chapter presented, the placement of SaaS components in cloud computing infrastructure. The CPP introduces new challenges and requirement constraints. To tackle these challenges in CPP, a heuristic algorithm SCPGA is applied to solve the CPP. The SCPGA evolves its solution as one population, which represents the identity of VM w.r.t. that particular component. The performance of SCPGA was tested in experiments with different number of VMs and SaaS components. The SCPGA was compared with the performance of First-fit RA. The evaluation focused on placement of the SaaS components without violating resource and SLA constraints. The results of experiments showed that the proposed SCPGA gives better performance in all set of experiments as compared with First-fit RA.

SCPGA also showed good scalability as problem size increases.

Chapter 4

Conclusions

4.1 Conclusions

In existing works, researchers proposed different placement policies for SaaS, which considered mainly some resource constraints like: processing capacity, memory and secondary storage. In recent work, researchers proposed a placement policy for SaaS, which considered SLA constraint for finding the cost of resources and profit of SaaS providers. Here, in this thesis revenue cost is also calculated. This thesis formulates Software-as-a-Service placement problem using service level agreement constraints and resource constraints. Our research developed SaaS Component Placement based on Genetic Algorithm for Software-as-a-Service placement on top of the virtual machines in Cloud computing infrastructure. The performance of SCPGA based on Software-as-a-Service placement policy has been compared with First-fit Randomized Algorithm. The experimental results verified that the proposed approach gives better results.

4.2 Future Work

CPP is an NP-hard problem. It is a broad area where very less work had to be done till now. There is more research required in this area. Our first future work is to design a general problem model considering more constraints like: number of cores, number of processors, and communication between SaaS components etc. And second future work is to design other heuristic frameworks to solve the offline CPP as well as online CPP.

Bibliography

[1] Francesco Maria Aymerich, Gianni Fenu, and Simone Surcis. An approach to a cloud computing network. In *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, pages 113–118. IEEE, 2008.

[2] Lee Badger, Tim Grance, Robert Patt-Corner, and Jeff Voas. Draft cloud computing synopsis and recommendations. *Recommendations of the National Institute of Standards and Technology*, 2011.

[3] Douglas K Barry. *Web Services, Service-oriented Architectures, and Cloud Computing: The Savvy Manager's Guide*. Access Online via Elsevier, 2012.

[4] Marina Berkovich, Sebastian Esch, Jan Marco Leimeister, and Helmut Krcmar. Towards requirements engineering for software as a service. *Multikonferenz Wirtschaftsinformatik 2010*, page 107, 2010.

[5] Robert B Bohn, John Messina, Fang Liu, Jin Tong, and Jian Mao. Nist cloud computing reference architecture. In *Services (SERVICES), 2011 IEEE World Congress on*, pages 594–596. IEEE, 2011.

[6] Lucinda Borovick and Rohit Mehra. Architecting the network for the cloud. 2011.

[7] Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*, pages 5–13. IEEE, 2008.

[8] K Selçuk Candan, Wen-Syan Li, Thomas Phan, and Minqi Zhou. At the frontiers of information and software as services. In *New Frontiers in Information and Software as Services*, pages 283–300. Springer, 2011.

[9] Erick Cantu-Paz. *Efficient and accurate parallel genetic algorithms*, volume 1. Springer, 2000.

[10] Frederick Chong and Gianpaolo Carraro. Architectures strategies for catching the long tail, 2006. *URL http://msdn.microsoft.com/en-us/library/aa479069.aspx*.

[11] CeCal High Performance Computing. Virtual machine planning in cloud computing systems. *URL http://www.fing.edu.uy/inco/grupos/cecal/hpc/VMMP/index.php*.

[12] Ron Condon. Forecast: Cloudy. *Computer methods in applied mechanics and engineering*, 1(3):15–19, 2011.

[13] Goolge Corporation. Google apps for business. *http://www.google.com/enterprise/apps/business/*.

[14] IBM Corporation. Ibm cloud. *URL http://www.ibm.com/cloud-computing/in/en/*.

[15] International Data Corporation. Saas revenue to grow five time faster than traditional packaged software through 2014,2010. *URL http://www.idc.com/about/viewpressrelease.jsp*.

[16] Microsoft Corporation. Microsoft office live small business. *URL http://www.smallbusiness.officelive.com*.

[17] Microsoft Corporation. Microsoft office live small business. *URL office.microsoft.com/en-IN*.

[18] Michael A Cusumano. The changing software business: Moving from products to services. *Computer*, 41(1):20–27, 2008.

[19] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 27–33. Ieee, 2010.

[20] Abhijit Dubey and Dilip Wagle. Delivering software as a service. *The McKinsey Quarterly*, 6(2007):2007, 2007.

[21] William Forrest and Charlie Barthold. Clearing the air on cloud computing. *Discussion Document from McKinsey and Company*, 2009.

[22] Armando Fox, Rean Griffith, A Joseph, R Katz, A Konwinski, G Lee, D Patterson, A Rabkin, and I Stoica. Above the clouds: A berkeley view of cloud computing. *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, 28:13, 2009.

[23] Borko Furht. Cloud computing fundamentals. In *Handbook of cloud computing*, pages 3–19. Springer, 2010.

[24] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[25] David Edward Goldberg et al. *Genetic algorithms in search, optimization, and machine learning*, volume 412. Addison-wesley Reading Menlo Park, 1989.

[26] M Hogan, F Liu, A Sokol, and J Tong. Nist cloud computing standards roadmap-version 1.0, special publication 500–291, december 2011, 2011.

[27] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[28] Anand V Hudli, Balasubrahmanya Shivaradhya, and Raghu V Hudli. Level-4 saas applications for healthcare industry. In *Proceedings of the 2nd bangalore annual compute conference*, page 19. ACM, 2009.

[29] Kai Hwang, Jack Dongarra, and Geoffrey C Fox. *Distributed and cloud computing: From parallel processing to the internet of things*. Morgan Kaufmann, 2013.

[30] Cisco System Inc. Cisco service-oriented network architecture: Support and optimise soa and web2.0 applications, 2008. *URL http://www.cisco.com/*.

[31] Gartner Inc. Gartner says worldwide software-as-a-service revenue to reach $14.5 billion in 2012,2012. *URL http://www.gartner.com/it/page.jsp?id=1963815*.

[32] Gartner Inc. Introducing saas-enabled application plateforms: Features, roles and futures, 2007.

[33] Salseforce.com Inc. Salseforce crm. *URL http://www.salesforce.com/in/service-cloud/overview*.

[34] Luit Infotech. Difference between the asp model and the saas model. *URL http://www.luitinfotech.com/kc/saas-asp-difference.pdf*.

[35] Thomas Plan K. Sel Chandan, Wen-Syan Li and Minqi Zhou. Frontiers in information and software as a service. In *In Proceeding of the IEEE 25th International Conference on Data Engineering*, pages 357–377. IEEE, 2010.

[36] Magnus Karlsson and Christos Karamanolis. Bounds on the replication cost for qos. Technical report, Citeseer, 2003.

[37] A Karve, Tracy Kimbrel, Giovanni Pacifici, Mike Spreitzer, Malgorzata Steinder, Maxim Sviridenko, and A Tantawi. Dynamic placement for clustered web applications. In *Proceedings of the 15th international conference on World Wide Web*, pages 595–604. ACM, 2006.

[38] Tatiana Kichkaylo. Timeless planning and the component placement problem. In *ICAPS Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[39] Tatiana Kichkaylo, Anca Ivan, and Vijay Karamcheti. Constrained component deployment in wide-area networks using ai planning techniques. In *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 10–pp. IEEE, 2003.

[40] Thomas Kwok and Ajay Mohindra. Resource calculations with constraints, and placement of tenants and instances for multi-tenant saas applications. In *Service-Oriented Computing–ICSOC 2008*, pages 633–648. Springer, 2008.

[41] Rosana SG Lanzelotte and Patrick Valduriez. Extending the search strategy in a query optimizer. In *VLDB*, pages 363–373, 1991.

[42] Phillip A Laplante, Jia Zhang, and Jeffrey Voas. Distinguishing between software oriented architecture and software as a service: What's in a name? *IEEE IT Professional*, 10(3):46–50, 2008.

[43] Zhipiao Liu, Shangguang Wang, Qibo Sun, Hua Zou, and Fangchun Yang. Cost-aware cloud service request scheduling for saas providers. *The Computer Journal*, 57(2):291–301, 2014.

[44] Thanasis Loukopoulos and Ishfaq Ahmad. Static and adaptive distributed data replication using genetic algorithms. *Journal of Parallel and Distributed Computing*, 64(11):1270–1285, 2004.

[45] P Mell and T Grance. Draft nist working definition of cloud computing-v15. *21. Aug 2009*, 2009.

[46] Hendrik Moens, Eddy Truyen, Stefan Walraven, Wouter Joosen, Bart Dhoedt, and Filip De Turck. Cost-effective feature placement of customizable multi-tenant applications in the cloud. *Journal of Network and Systems Management*, pages 1–42, 2013.

[47] Marek Obitko. Introduction to genetic algorithms. *URL http://www.obitko.com/tutorials/genetic-algorithms/parameters.php*.

[48] Pankesh Patel, Ajith H Ranabahu, and Amit P Sheth. Service level agreement in cloud computing. 2009.

[49] Colin R Reeves. A genetic algorithm for flowshop sequencing. *Computers & operations research*, 22(1):5–13, 1995.

[50] Wei-Tek Tsai, Xin Sun, and Janaka Balasooriya. Service-oriented cloud computing architecture. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 684–689. IEEE, 2010.

[51] Mark Turner, David Budgen, and Pearl Brereton. Turning software into a service. *Computer.*, 36(10):38–44, 2003.

[52] Bhuvan Urgaonkar, Arnold L Rosenberg, and Prashant Shenoy. Application placement on a cluster of servers. *International Journal of Foundations of Computer Science*, 18(05):1023–1041, 2007.

[53] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[54] Toby Velte, Anthony Velte, and Robert Elsenpeter. *Cloud computing, a practical approach*. McGraw-Hill, Inc., 2009.

[55] Xiaolong Yang and Huimin Zhang. Cloud computing and soa convergence research. In *Computational Intelligence and Design (ISCID), 2012 Fifth International Symposium on*, volume 1, pages 330–335. IEEE, 2012.

[56] Mohd Yusoh and Zeratul Izzah. Composite saas resource management in cloud computing using evolutionary computation. 2013.

[57] Zeratul Izzah Mohd Yusoh and Maolin Tang. A penalty-based genetic algorithm for the composite saas placement problem in the cloud. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–8. IEEE, 2010.

[58] Zeratul Izzah Mohd Yusoh and Maolin Tang. Composite saas placement and resource optimization in cloud computing using evolutionary algorithms. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 590–597. IEEE, 2012.

[59] Longwen Zhao and Jinyu Liu. Component-oriented saas integration framework research based on ofbiz. In *Management and Service Science (MASS), 2011 International Conference on*, pages 1–3. IEEE, 2011.

[60] Minqi Zhou, Rong Zhang, Dadan Zeng, and Weining Qian. Services in the cloud computing era: A survey. In *Universal Communication Symposium (IUCS), 2010 4th International*, pages 40–46. IEEE, 2010.

[61] Xiaoyun Zhu, Cipriano Santos, Dirk Beyer, Julie Ward, and Sharad Singhal. Automated application component placement in data centers using mathematical programming. *International journal of network management*, 18(6):467–483, 2008.

[62] Barbora Zimmerova et al. Component placement in distributed environment wrt component interaction. In *Proceedings of the 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 260–267, 2006.